Apuntes de Lenguajes Formales

Leonardo H. Añez Vladimirovna[1]

Universidad Autónoma Gabriél René Moreno,
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones,
Santa Cruz de la Sierra, Bolivia

12 de noviembre de 2019

[1]Correo Electrónico: toborochi98@outlook.com

Agradecimiento a marmot

Notas del Autor

Estos apuntes fueron realizados durante mis clases en la materia `INF319 (Lenguajes Formales)`, acompañados de referencias de libros, fuentes y código que use a lo largo del curso, en el período `I-2019` en la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones.

Agradecimiento(s):

- Mauricio Elian Delgadillo García

Para cualquier cambio, observación y/o sugerencia pueden enviarme un mensaje al siguiente correo:

`toborochi98@outlook.com`

Índice general

Capítulo 1

Preliminares Formales

1.1. Conjuntos

1.1.1. Conjunto Finito e Infinito

Equivalencia

Dado A y B (conjuntos) los llamamos *equivalentes* si existe una biyección: $f : A \to B$

Conjunto Finito

Un conjunto A es finito si es equivalente a $\{1, 2, 3, \ldots, n\}$ para algún $n \in \mathbb{N}$.

Conjunto Infinito

Un conjunto es infinito si no es finito. Si no es equivalente a $\{1, 2, 3, \ldots, n\}$ es decir no hay biyección. Sin embargo no todos los conjuntos finitos son equivalentes.

- **Conjunto Contablemente Infinito:** Se dice que un conjunto es contablemente infinito si es equivalente con $\mathbb{N}$.
- **Conjunto Contable:** Es contable si es finito o contablemente infinito.
- **Conjunto Incontable:** Se dice que es incontable si no es contable.

Principio de las Casillas

Si A y B son conjuntos finitos no vacíos y $|A| > |B|$ entonces no existe una función inyectiva de: $A \to B$.

1.2. Preliminares

1.2.1. Alfabeto

Un alfabeto Σ es cualquier conjunto finito no vacío.

Ejemplo(s)

$$\begin{aligned}\Sigma_1 &= \{Leo, Martha\}\\ \Sigma_2 &= \{0, 1, 2, 3, \ldots, 13\}\\ \Sigma_3 &= \{a, b\}\\ \Sigma_4 &= \{R, G, B, A\}\end{aligned}$$

1.2.2. Palabra

Una palabra sobre Σ es una sucesión finita de símbolos de Σ. Es decir:

$$(\sigma_1, \sigma_2, \ldots, \sigma_n); \sigma \in \Sigma \qquad \text{ó} \qquad \sigma_1\sigma_2\sigma_3 \ldots \sigma_n; \sigma \in \Sigma$$

Ejemplo(s)

Sobre Σ_1	**Sobre** Σ_2	**Sobre** Σ_3	**Sobre** Σ_4
$w_1 = LeoLeo$	$w_1 = 1111110$	$w_1 = bababababa$	$w_1 = ABGR$
$w_2 = MarthaLeoMartha$	$w_2 = 11235813$	$w_2 = abba$	$w_2 = RRRA$

Denotamos por Σ^* el conjunto de todas las palabras sobre Σ.

Longitud de una Palabra

Sea w una palabra sobre Σ, es decir $w = \sigma_1\sigma_2 \dots \sigma_n; \sigma \in \Sigma$. La longitud de w es n y se denota por: $|w| = n$.

Palabra vacía

Es la sucesión vacía de símbolos de Σ y se denota por: λ.

1.2.3. Notaciones

- $\Sigma^+ = \{w \in \Sigma^* / |w| > 0\}$
- $\Sigma^k = \{w \in \Sigma^* / |w| = k\}$
- $\Sigma^0 = \{w \in \Sigma^* / |w| = 0\} = \{\lambda\}$
- $\Sigma^1 = \{w \in \Sigma^* / |w| = 1\} = \Sigma$
- $\Sigma^* = \Sigma^+ \cup \{\lambda\}$
- $\Sigma^+ = \Sigma^* - \{\lambda\}$

1.2.4. Cantidad de Ocurrencias

Sea $w \in \Sigma^*$, denotamos por $|w|_\sigma$ al número de ocurrencias del símbolo σ en la palabra w.

Ejemplo(s)

$$\Sigma = \{a, b\}$$

- $\Sigma^* = \{\lambda, a, b, aa, bb, ab, ba, aaa, \dots\}$
- $\Sigma^0 = \{\lambda\}$
- $\Sigma_1 = \Sigma = \{a, b\}$
- $\Sigma^2 = \{ab, aa, ba, bb\}$

1.2.5. Concatenación

Sea $u, v \in \Sigma^*$ tal que $u = \sigma_1\sigma_2 \dots \sigma_n, v = \epsilon_1\epsilon_2 \dots \epsilon_n$. La concatenación de u y v se define por:

$$uv = \sigma_1\sigma_2 \dots \sigma_n\epsilon_1\epsilon_2 \dots \epsilon_n$$

Propiedades

- $uv \neq vu$
- $(uv)w = u(vw)$
- $u\lambda = \lambda u = u$

Ejemplo(s)

$$u = abab \qquad uv = ababbba$$
$$v = bba \qquad vu = bbaabab$$

1.2.6. Cardinalidad

Definición de Recurrencia

$$|\ | : \Sigma^* \to \mathbb{N}$$
$$\begin{cases} |\lambda| = 0 \\ |wa| = |w| + 1 \end{cases}$$

Propiedades

- $|uv| = |u| + |v|$
- $|uv|_a = |u|_a + |v|_a$

Ejemplo(s)

- $v = abab, |v| = 4$

1.2.7. Inversa

Si $w = \sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma^n$ entonces $w' = \sigma_n, \sigma_{n-1}, \ldots, \sigma_1$ se llama inversa o transpuesta de w.

Definición de Recurrencia

$$' : \Sigma^* \to \Sigma^*$$
$$\begin{cases} \lambda' = \lambda \\ (wa)' = aw' \end{cases}$$

Ejemplo(s)

- Demostrar $|w| = |w'|; \forall w \in \Sigma^*$
  Prueba

$$L = \{w \in \Sigma^* / |w| = |w'|\}$$

a) ¿$\lambda \in L$?

$$\lambda = \lambda' \Rightarrow |\lambda| = 0 = |\lambda'|$$

i) $\therefore \lambda \in L$

b) ¿$w \in L$?

p.d. $w \in L \wedge a \in \Sigma \Rightarrow wa \in L$
Sea

$$w \in L \wedge a \in \Sigma$$
$$|w| = |w'| \wedge a \in \Sigma \ \textbf{HI}$$

p.d. $wa \in L$
p.d. $|(wa)'| = |aw'|$

$$\begin{aligned} |(wa)'| &= |aw'| \\ &= |a| + |w'| \\ &= |a| + |w| \\ &= |w| + |w| \\ &= |wa| \end{aligned}$$

ii) $\therefore w \in L \wedge a \in \Sigma \Rightarrow wa \in L$

Entonces: $L = \Sigma^*$

1.2.8. Potencia de una Palabra

$$w^n = \underbrace{ww \dots w}_{n-veces}$$

Definición de Recurrencia

$$^n : \Sigma^* \to \Sigma^*$$

$$w^1 = w$$

$$\begin{cases} w^0 = \lambda \\ w^{n+1} = ww^n \end{cases}$$

Propiedades

- $|w^n| = n|w|$
- $w^m w^n = w^{m+n}$
- $(w^n)^m = w^{mn}$
- $\lambda^n = \lambda$

1.2.9. Principio de Inducción para Σ^*

Sea L un conjunto de palabras sobre Σ con las propiedades:

i.) $\lambda \in L$

ii.) $w \in L \wedge a \in \Sigma \Rightarrow wa \in L$

Crecimiento por Izquierda

ii') $w \in L \wedge a \in \Sigma \Rightarrow aw \in L$

Entonces

$L = \Sigma^*$, (es decir, todas las palabras sobre Σ están en L.)

Ejemplo(s)

- Demostrar: $|uv| = |u| + |v| \ \forall u, v \in \Sigma^*$
  Prueba

$$L = \{v \in \Sigma^* / |uv| = |u| + |v|\}$$

a) ¿$\lambda \in L$?

$$|u\lambda| = |u| = |u| + 0 = |u| + |\lambda|$$

i) $\therefore \lambda \in L$

b) ¿$w \in L$?

p.d. $w \in L \wedge a \in \Sigma \Rightarrow wa \in L$
Sea

$$w \in L \wedge a \in \Sigma$$

$$|uw| = |u| + |w| \wedge a \in \Sigma \ \textbf{HI}$$

p.d. $wa \in L$
p.d. $|u(wa)| = |u| + |wa|$

$$\begin{aligned} |u(wa)| &= |(uw)a| \\ &= |uw| + 1 \\ &= |u| + |w| + 1 \\ &= |u| + (|w| + 1) \\ &= |u| + |wa| \end{aligned}$$

ii) $\therefore w \in L \wedge a \in \Sigma \Rightarrow wa \in L$

Entonces: $L = \Sigma^*$

1.2.10. Prefijos, Sufijos y Subpalabras

Sean $v, z \in \Sigma^*$:

- Se dice que v es prefijo de z **ssi** existe $w \in \Sigma^*$ tal que: $z = vw$. Y se escribe: "v *pref* z" .
- Se dice que v es sufijo de z **ssi** existe $u \in \Sigma^*$ tal que: $z = uv$. Y se escribe: "v *suf* z" .
- Se dice que v es subpalabra de z **ssi** existe $u_1, u_2 \in \Sigma^*$ tal que: $z = u_1 \vee u_2$. Y se escribe: "v *subpz*" .

Sean $x, w \in \Sigma^*$:

- x es **prefijo** de w si $w = xy$, si $\exists y \in \Sigma^* / w = xy$
- x es **sufijo** de w si $w = yx$, si $\exists y \in \Sigma^* / w = yx$
- x es **subcadena** de w si $w = yx$, si $\exists z, y \in \Sigma^* / w = zxy$

Ejemplo(s)

- $\Sigma = \{a, b\}$, $z = babaab$

Prefijo		**Sufijo**		**Subpalabra**		
$u_1 = ba$	$w = baab$	$v_1 = aab$	$u = bab$	$s_1 = aa$	$u_1 = bab$	$u_2 = b$
$u_2 = bab$	$w = aab$	$v_2 = abaab$	$u = b$	$s_2 = baa$	$u_1 = ba$	$u_2 = b$
$u_3 = baba$	$w = ab$	$v_3 = ab$	$u = baba$	$s_3 = abaa$	$u_1 = b$	$u_2 = b$
$u_4 = babaa$	$w = b$	$v_4 = baab$	$u = ba$	$s_4 = ba$	$u_1 = ba$	$u_2 = ab$
$u_5 = babaab$	$w = \lambda$	$v_5 = b$	$u = babaa$	$s_5 = b$	$u_1 = \lambda$	$u_2 = abaab$

1.2.11. Lenguajes

Un lenguaje sobre Σ es un subconjunto de Σ^*

Ejemplo(s)

- $\emptyset \subseteq A; \forall A$
- $A \subseteq A; \forall A$
- $\Sigma = \{a, b\}, \Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, \ldots\}$ Podemos tener los siguientes lenguajes:
  - $L_1 = \{baba, aba, abba, bb, bababababababababababababa\}$
  - $L_2 = \{\lambda, a, b\}$
  - $L_3 = \{aaa, abb, bbb\}$
  - $L_4 = \{w \in \Sigma^* / |w| \leq 5 \wedge |w|_a - |w|_b = 1\}$
  - $L_5 = \{aab, aaabb, ababa, \ldots, aba\}$
  - $L_6 = \{a, ab, abb, abbb, \ldots\}$

Operaciones

Recordemos que ya conocemos otras operaciones (Unión, Intersección, Diferencia y Complemento), para esta asignatura tenemos las siguientes:

- **Concatenación**

  Sea $A, B \subseteq \Sigma^*$

  $$AB = \{w \in \Sigma^* / w = xy, x \in A, y \in B\}$$

- **Transposición**

  Sea $A \subseteq \Sigma^*$

  $$A' = \{w' \in \Sigma^* / w \in A\}$$

- **Estrella de Kleene**

  Sea $A \subseteq \Sigma^*$

  $$A^* = \{w \in \Sigma^* / w = w_1 w_2 \dots w_n \text{ para algún } k \in \mathbb{N} \text{ y para algunas } w_1, w_2, \dots, w_k \in A\}$$

Ejemplo(s)

- Dado $A = \{b, ab\}, B = \{ba, bab\}$ realizar:
  a) $A' = \{b, ba\}$
  b) $B' = \{ab, bab\}$
  c) $AB = \{bba, bbab, abba, abbab\}$
  d) $A^* = \{\lambda, b, ab, bb, bab, abb, abab, bbb, babb, bbab, \dots\}$
- Sea $\Sigma = \{a, b\}$ y sean $P = \{a, ab\}, Q = \{\lambda, a, ba\}$ realizar:
  a) $P \cup Q = \{a, ab, ba, \lambda\}$
  b) $P \cap Q = \{a\}$
  c) $P - Q = \{ab\}$
  d) $Q - P = \{\lambda, ba\}$
  e) $P' = \{a, ba\}$
  f) $Q' = \{\lambda, a, ab\}$
  g) $PQ = \{a, aa, aba, abba\}$
  h) $P^2 = PP = \{aa, aab, aba, abab\}$

1.2.12. Orden Lexicográfico

Cualquier lenguaje infinito puede ser ordenado lexicograficamente en consecuencia se puede poner en biyección con el conjunto $\mathbb{N}$.

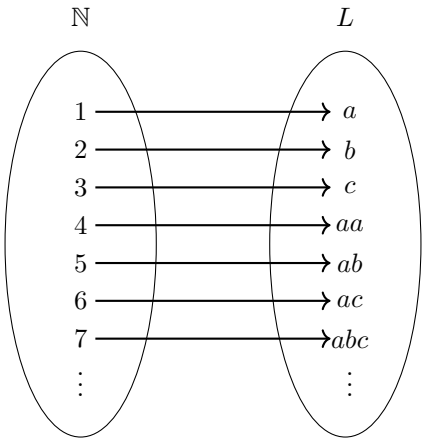
i.) Se presupone un orden en los símbolos del alfabeto.

ii.) Entre dos palabras de distinta longitud la de menor longitud precede a la de mayor longitud.

iii.) Entre 2 palabras de la misma longitud se factoriza ambas hasta el primer símbolo diferente exclusivo. La precedencia se decide comparando la segunda parte de ambas factorizaciones, en base al primer símbolo de la segunda parte.

Ejemplo(s)

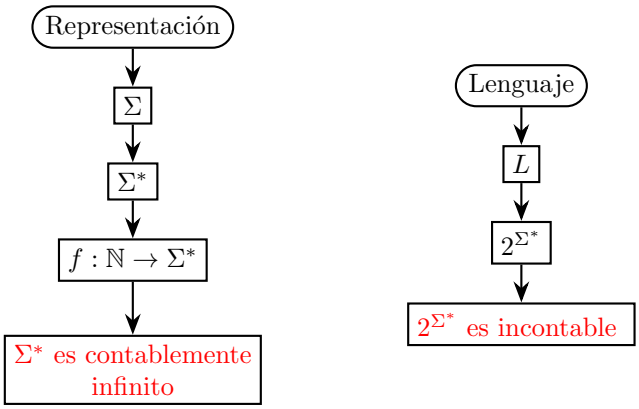
- $\Sigma = \{a, b, c\},\ L = \{a, b, ab, ac, acb, abc, c, aa, cba\}$



1.2.13. Representación Finita de Lenguajes

Necesitamos tener dos consideraciones:

- Cualquier representación, debe ser en si misma una palabra. Es decir una sucesión finita de símbolos de un alfabeto Σ.
- Lenguajes diferentes deben tener representaciones diferentes.



Teorema

Conjunto $2^{\mathbb{N}}$ es incontable:

Prueba (R.A.A.)
Supongamos que $2^{\mathbb{N}}$ es contable.
$2^{\mathbb{N}}$ es contablemente infinito.

Es decir que existe $f : \mathbb{N} \to 2^{\mathbb{N}} / f(i) = S_i$ para algún $i \in \mathbb{N}$ construimos un conjunto:

$$D = \{n \in \mathbb{N} / n \notin S_n\}$$

D es un subconjunto de $\mathbb{N}$ entonces $D = S_k$,para algún $k \in \mathbb{N}$
Por lo tanto, la pregunta: ¿$k \in S_n$?

$$k \notin S_k \Rightarrow k \in D \Rightarrow k \in S_k \Rightarrow\Leftarrow$$
$$k \in S_k \Rightarrow k \in D \Rightarrow k \notin S_k \Rightarrow\Leftarrow$$

$\therefore 2^{\mathbb{N}}$ es incontable.

Primer resultado de la Teoría de la Computación

No importa cuán poderosos sean los métodos que conocemos para representar lenguajes, sólo pueden ser representados un número contable de ellos (mientras que las representaciones sean finitas), inevitablemente, un número incontable de ellos quedará sin representación bajo cualquier esquema de representación finita.

1.2.14. Expresiones Regulares

Las expresiones regulares (ER) sobre un alfabeto (Σ) son las palabras sobre el alfabeto $\Sigma \cup \{), (, \emptyset, \cup, *\}$ tal que cumple lo siguiente:

1.) $\emptyset$ y cada símbolo de Σ es una ER.

2.) Si α y β son ER entonces $(\alpha\beta)$ es una ER.

3.) Si α y β son ER entonces $(\alpha \cup \beta)$ es una ER.

4.) Si α es una ER entonces α^* es una ER.

5.) Nada mas es una ER a menos que provenga de **(1.)** a **(4.)**

Ejemplo(s)

- Para $\Sigma = \{a, b\}$ podemos formar:

$$\begin{aligned}
&\emptyset \\
&a \\
&b \\
&(ab) \\
&(a \cup b) \\
&((ab) \cup b) \\
&(ba)^* \\
&((ba)^* \cup (a \cup b))^*
\end{aligned}$$

- Escribir por comprensión el lenguaje generado por la expresión: $\alpha = a^*$.

$$\begin{aligned}
L(\alpha) &= L(a^*) = L(a)^* \\
&= \{a\}^* \\
&= \{\lambda, a, aa, aaa, \ldots\} \\
&= \{w \in \Sigma^* / w = a^n, n \geq 0\}
\end{aligned}$$

Lenguaje Regular

Un lenguaje es regular ssi es generado por una expresión regular.

Ejemplo(s)

- Dado: $\Sigma = \{a, b\}$. Escribir por comprensión los lenguajes generados por las *ER*:

  a) $\alpha = ab^*$

$$\begin{aligned}
L(\alpha) &= L(ab^*) = L(a)L(b^*) = L(a)L(b)^* \\
&= \{a\}\{b\}^* \\
&= \{a\}\{\lambda, b, bb, bbb, \ldots\} \\
&= \{a, ab, abb, abb, \ldots\} \\
L(\alpha) &= \{w \in \Sigma^* / w = ab^n, n \geq 0\}
\end{aligned}$$

  b) $\alpha = ba^*b$

$$\begin{aligned}
L(\alpha) &= L(ba^*b) \\
&= L(ba^*)L(b) \\
&= L(b)L(a^*)L(b) \\
&= \{b\}\{\lambda, a, aa, aaa, \ldots\}\{b\} \\
&= \{b, bab, baab, baaab, \ldots\} \\
L(\alpha) &= \{w \in \Sigma^* / w = ba^nb, n \geq 0\}
\end{aligned}$$

$\cup$ significa selección: $a \cup ab$ significa, "escojo a o escojo ab."

c) $\alpha = (a \cup b)^* a$

$$\begin{aligned} L(\alpha) &= L((a \cup b)^*)L(a) \\ &= L((a \cup b))^* L(a) \\ &= (L(a) \cup L(b))^* L(a) \\ &= (\{a\} \cup \{b\})^* \{a\} \\ &= \{a, b\}^* \{a\} \\ &= \underbrace{\{\lambda, a, b, ab, aa, ba, bb, \ldots\}}_{\Sigma^*} \{a\} \end{aligned}$$

$$L(\alpha) = \{w \in \Sigma^* / w = va \wedge v \in \Sigma^*\}$$

d) $\alpha = (a \cup b)^* a (a \cup b)^*$

$$\begin{aligned} L(\alpha) &= L((a \cup b)^* a (a \cup b)^*) \\ &= L((a \cup b))^* L(a) L((a \cup b))^* \\ &= (\{a\} \cup \{b\})^* \{a\} (\{a\} \cup \{b\})^* \\ &= \{a, b\}^* \{a\} \{a, b\}^* \\ &= \underbrace{\{\lambda, a, b, aa, ab, bb, ba, \ldots\}}_{\Sigma^*} \{a\} \underbrace{\{\lambda, a, b, aa, ab, bb, ba, \ldots\}}_{\Sigma^*} \end{aligned}$$

$$L(\alpha) = \{w \in \Sigma^* / w = xax, x \in \Sigma^*\}$$

e) $\alpha = b^* a b^*$

$$\begin{aligned} L(\alpha) &= L(b^* a b^*) \\ &= L(b^*) L(a) L(b^*) \\ &= \{b\}^* \{a\} \{b\}^* \\ &= \{\lambda, b, bb, bbb, \ldots\} \{a\} \{\lambda, b, bb, bbb, \ldots\} \\ &= \{a, bab, bbabb, bbbabbb, \ldots\} \end{aligned}$$

$$L(\alpha) = \{w \in \Sigma^* / w = b^n a b^n, n \geq 0\}$$

1.2.15. Módulos

Antes de comenzar con las formalidades de un módulo, es bastante práctico contar con una introducción simple. Imaginemos que tenemos una correa transportadora controlada por una llave:

Movimiento de la Correa

- Parada
- Derecha
- Izquierda

Posición de la Llave

- Superior
- Inferior

Su funcionamiento estaría dado de la siguiente manera:

Correa \ Llave	Inferior	Superior
Parada	Izquierda	Derecha
Derecha	Izquierda	Derecha
Izquierda	Parada	Derecha

El **movimiento** realizado por la correo estaría dado por el siguiente conjunto:

$$k = \{p, i, d\} \text{ siendo } p, i, d \text{ parada,izquierda,derecha respectivamente.}$$

Y los **estados** de la llave serían:

$$\Sigma = \{+, -\} \text{ siendo } +, - \text{ superior, inferior.}$$

El funcionamiento podríamos verlo como una función definida de la siguiente manera: $f : k \times \Sigma \to k$
Donde:

- $k \times \Sigma = \{(p, +), (p, -), (i, +), (i, -), (d, +), (d, -)\}$

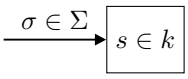
Definición

Un módulo es una tripleta $D = (k, \Sigma, f)$ donde:

- k es un conjunto finito no vacío, *llamado conjunto de estados*
- Σ es un conjunto finito no vacío, *llamado alfabeto*
- $f : k \times \Sigma \to k$, *llamado función de transición*

Interpretación

Un módulo se puede interpretar como un dispositivo que en determinados instantes de tiempo recibe señales (símbolos del alfabeto), que producen cambios en su configuración interna.



Representación

- **Tabla de Transición**

k \ Σ	σ_0	σ_1	$\cdots$	σ_j	$\cdots$	σ_m
s_1				$\vdots$		
s_2				$\vdots$		
$\vdots$				$\vdots$		
s_i	$\cdots$	$\cdots$	$\cdots$	s_k		
$\vdots$						
s_n						

Figura 1.1: $s_k = f(s_i, \sigma_j)$

- **Grafo**

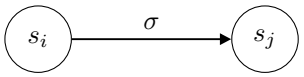


Figura 1.2: ssi: $f(s_i, \sigma) = s_j$

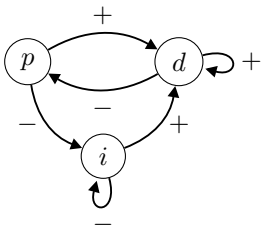
Ejemplo(s)

- Para el anterior ejemplo:

  a) Tabla de transición:

k \ Σ	$+$	$-$
p	d	i
i	d	i
d	d	p

  b) Grafo:



Comportamiento Dinámico

Sea $D = (k, \sigma, f)$ un módulo:

$$\begin{array}{ccccccccc} t_0 & & t_1 & & t_2 & & \cdots & & t_k \\ s_0 & \xrightarrow{\sigma_0} & s_1 & \xrightarrow{\sigma_1} & s_2 & \xrightarrow{\sigma_2} & \cdots & \xrightarrow{\sigma_{k-1}} & s_k \end{array}$$

$$s_{i+1} = f(s_i, \sigma_i) \text{ Procesa Símbolo}$$
$$s_k = \hat{f}(s_0, \sigma_0\sigma_1\sigma_2 \ldots \sigma_{k-1}) \text{ Procesa Palabras}$$

Función Estado Terminal

Sea $D = (k, \sigma, f)$ un módulo:

Una función de Estado Terminal del módulo D es una única función:

$$\widehat{f} : k \times \Sigma \to k \text{ tal que } \forall s \in k, w \in \Sigma^*, \sigma \in \Sigma$$

$$\begin{cases} \widehat{f}(s, \lambda) = s \\ \widehat{f}(s, \sigma w) = \widehat{f}\,[f(s, \sigma), w] \end{cases}$$

◇ **Notas**

- $w = \lambda$

$$\widehat{f}(s, \sigma) = \widehat{f}(s, \sigma\lambda) = \widehat{f}\,[f(s, \sigma), \lambda] = f(s, \sigma)$$

- $\forall w \in \Sigma^*$

$$f : k \to k \text{ tal que: } f_w(s) = \widehat{f}(s, w)s$$

Ejemplo(s)

- Digamos que tenemos un edificio de 3 plantas y este posee un ascensor:
  Donde:
  - p, s, t se refiere a primero, segundo, tercero respectivamente
  - $-, +$ se refiere a bajar y subir, respectivamente.

k \ Σ	$-$	$+$
p	p	s
s	p	t
t	s	t

Procesar $\hat{f}(s, --++)$. Podemos procesarlo de la siguiente manera:

$$s \xrightarrow{-} p \xrightarrow{-} p \xrightarrow{+} s \xrightarrow{+} t$$

o bien, de una manera mas formal:

$$\begin{aligned}\hat{f}(s,--++) &= \hat{f}[f(s,-),-++] \\ &= \hat{f}(p,-++) \\ &= \hat{f}[f(p,-),++] \\ &= \hat{f}(p,++) \\ &= \hat{f}[f(p,+),+] \\ &= \hat{f}(s,+) = f(s,+) = t\end{aligned}$$

1.2.16. Alcanzabilidad

Sea $D = (k, \Sigma, f)$ un módulo y sean $s, t \in K$.

- Decimos que t es alcanzable a partir de s **ssi** existe $w \in \Sigma^*$ tal que:

$$\hat{f}(s, w) = t$$

y se escribe

$$s \longrightarrow t$$

- Sea $k \in \mathbb{N}$. Decimos que t es k-Alcanzable a partir de s **ssi** existe $w \in \Sigma^*$ tal que

$$|w| = k \wedge \hat{f}(s, w) = t$$

y se escribe:

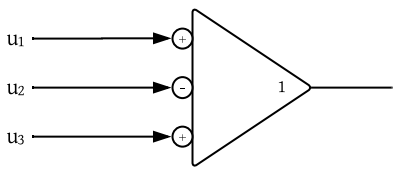
$$s \xrightarrow{k} t$$

◇ **Notas**

La alcanzabilidad no implica k alcanzabilidad
La k alcanzabilidad implica alcanzabilidad

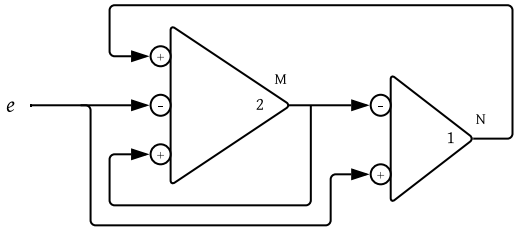
Ejemplo(s)

- Tengamos el ejemplo de una *neurona*.
  Donde:
  - $\oplus$: Excitación
  - $\ominus$: Inhibición



$$S(k+1) = \begin{cases} 1 \text{ si } u_1(k) + u_3(k) - u_2(k) \geq 1 \\ 0 \textbf{ e.o.c.} \end{cases}$$

k \ Σ	$(0,0,0)$	$(0,0,1)$	$(0,1,0)$	$(0,1,1)$	$(1,0,0)$	$(1,0,1)$	$(1,1,0)$	$(1,1,1)$
1	0	1	0	0	1	1	0	1
0	0	1	0	0	1	1	0	1



- Veamos con dos *neuronas*: Denotaciones $s(i) \to M, t(i) \to N, e(i)$

$$S(k+1) = \begin{cases} 1 & \text{si } t(k) + s(k) - e(k) \geq 2 \\ 0 & \textbf{e.o.c.} \end{cases}$$

$$t(k+1) = \begin{cases} 1 & e(k) - s(k) \geq 1 \\ 0 & \textbf{e.o.c.} \end{cases}$$

k \ Σ	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
1	$(0,1)$	$(0,1)$	$(0,0)$	$(0,0)$
0	$(0,0)$	$(0,0)$	$(0,0)$	$(1,0)$

1.2.17. Máquinas

Al igual que la sección de Módulos (*1.2.15*) sería bueno tener un ejemplo previo a las definiciones formales. Digamos que dos personas lanzan una moneda repetidamente. En cada instante, el dispositivo n:

- Recibe uno de los símbolos a (simbolizando cara) o b (simbolizando cruz).
- Emite un símbolo que puede ser 1 o 2 indicando respectivamente que el primero o segundo jugador recibe un punto.
- Puede estar en uno de los siguientes tres estados:
  - c: Indicando que el lanzamiento anterior fue cara.
  - p: Indicando que la cruz que salió en el lanzamiento anterior fue la primera de una sucesión de cruces.
  - d: Indicando que ya salieron 2 o mas cruces en la sucesión actual de cruces.

El funcionamiento del sistema será el siguiente:

- Recibiendo el símbolo a pasará al estado c, independientemente del estado actual y emitirá el 2 si estaba en el estado c y 1 en caso contrario.
- Recibiendo el símbolo b emitirá el símbolo 1, quedando en el mismo estado si estaba en d, caso contrario la salida será 2 y si estaba en c pasará a p, si estaba en p pasará a d.

El funcionamiento puede ser descrito por dos funciones:

$$f\begin{pmatrix} Estado \\ Actual, \end{pmatrix}\begin{matrix} Simbolo \\ Leido \end{matrix} = \begin{matrix} Proximo \\ Estado \end{matrix} \qquad g\begin{pmatrix} Estado \\ Actual, \end{pmatrix}\begin{matrix} Simbolo \\ Leido \end{matrix} = \begin{matrix} Salida \\ Emitida \end{matrix}$$

Ademas tenemos los conjuntos que definen estos estados y transiciones:

$$k = \{c, p, d\}, \Sigma = \{a, b\}, k \times \Sigma = \{(c,a), (c,b), (p,a), (p,b), (d,a), (d,b)\}$$

Por lo que finalmente el dispositivo n estaría descrito de la siguiente manera:

$$\begin{array}{ll} f : k \times \Sigma \to k & g : k \times \Sigma \to \Delta \\ f(c,a) = c & g(c,a) = 2 \\ f(c,b) = p & g(c,b) = 2 \\ f(p,a) = c & g(p,a) = 1 \\ f(p,b) = d & g(p,b) = 2 \\ f(d,a) = c & g(d,a) = 1 \\ f(d,b) = d & g(d,a) = 1 \end{array}$$

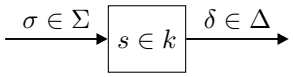
Definición

Una máquina es una quíntupla $M = (k, \Sigma, \Delta, f, g)$ donde:

- k es un conjunto finito no vacío, *llamado conjunto de estados*
- Σ es un conjunto finito no vacío, *llamado alfabeto de entrada*
- Δ es un conjunto finito no vacío, *llamado alfabeto de salida*
- $f : k \times \Sigma \to k$, *llamado función de transición*
- $g : k \times \Sigma \to \Delta$, *llamado función de salida*

Interpretación

Una máquina se puede interpretar como un dispositivo que en determinados instantes de tiempo recibe señales (símbolos de entrada) que producen cambios en su configuración interna y emiten señales (símbolos de salida).



Representación

- **Tabla de Transición**

k \ Σ	σ_0	σ_1	$\cdots$	σ_j	$\cdots$	σ_m
s_1				$\vdots$		
s_2				$\vdots$		
$\vdots$				$\vdots$		
s_i	$\cdots$	$\cdots$	$\cdots$	s_k/δ_k		
$\vdots$						
s_n						

Figura 1.3: $s_k = f(s_i, \sigma_j) \quad g(s_1, \sigma_j) = \delta_k$

- **Grafo**

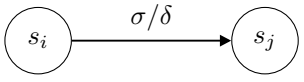


Figura 1.4: ssi: $f(s_i, \sigma) = s_j \quad g(s_i, \sigma) = \delta$

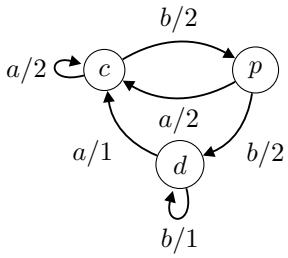
Ejemplo(s)

La representación del ejemplo anterior:

a) Tabla

k \ Σ	a	b
c	$c/2$	$p/2$
p	$c/1$	$d/2$
d	$c/1$	$d/1$

b) Grafo



Comportamiento Dinámico

Sea $M = (k, \Sigma, \Delta, f, g)$ una máquina:

$$\begin{array}{ccccccccc} t_0 & & t_1 & & t_2 & & \cdots & & t_k \\ s_0 & \xrightarrow{\sigma_0/\delta_0} & s_1 & \xrightarrow{\sigma_1/\delta_1} & s_2 & \xrightarrow{\sigma_2/\delta_2} & \cdots & \xrightarrow{\sigma_{k-1}/\delta_{k-1}} & s_k \end{array}$$

$$\begin{aligned} \hat{f}(s_0, \sigma_0\sigma_2 \ldots \sigma_{k-1}) &= s_k \\ y(s_i, \sigma_i) &= \delta_i \\ \overline{y}(s_0, \sigma_0\sigma_1 \ldots \sigma_k) &= \delta_0\delta_1 \ldots \delta_{k-1} \end{aligned}$$

Función Estado Terminal

Sea $M = (k, , \Sigma, \Delta, f, g)$ una máquina.

a) Una función de Estado Terminal de M es la función de estado terminal:

$$\widehat{f} : k \times \Sigma^* \to k \text{ del módulo } (k, \Sigma, f)$$

b) Una función palabra de salida de M es una única función:

$$\overline{g} : k \times \Sigma^* \to \Delta^* \text{ tal que } \forall s \in k, \sigma \in \Sigma, w \in \Sigma^*$$

$$\begin{cases} \overline{g}(s, \lambda) & = \lambda \\ \overline{g}(s, \sigma w) & = g(s, \sigma)\overline{g}\big[f(s, \sigma), w\big] \end{cases}$$

◇ **Notas**

- $w = \lambda$

$$\overline{g}(s, \sigma) = \overline{g}(s, \sigma\lambda) = g(s, \sigma)\underbrace{\overline{g}\big[f(s, \sigma), \lambda\big]}_{\lambda} = g(s, \sigma)\lambda = g(s, \sigma)$$

- $\forall s \in k$

$$g_s : \Sigma^* \to \Delta^* \text{ tal que } g_s(w) = \overline{g}(s, w)$$

1.2.18. Síntesis, Análisis y Verificación

	I/O	Máquina
Síntesis	✓	?
Análisis	?	✓
Verificación	✓	✓

Capítulo 2

Autómatas

2.1. Autómata Finito Determinístico (AFD)

2.1.1. Definición

Un Autómata Finito Determinístico (AFD) es una quintupla $M = (k, \Sigma, f, s_0, F)$ donde:

- k conjunto finito no vacio, *conjunto de estados*
- Σ conjunto finito no vacio, *Alfabeto*
- $f : k \times \Sigma \to k$, *Funcion de transicion*
- $s_0 \in k$, *Estado inicial*
- $F \subseteq k$, *Conjunto de estados finales*

2.1.2. Intepretación

Podemos verlo como una cinta que posee una **cabeza lectora**, esta se mueve hacia la derecha y con este movimiento los estados q_i cambian.

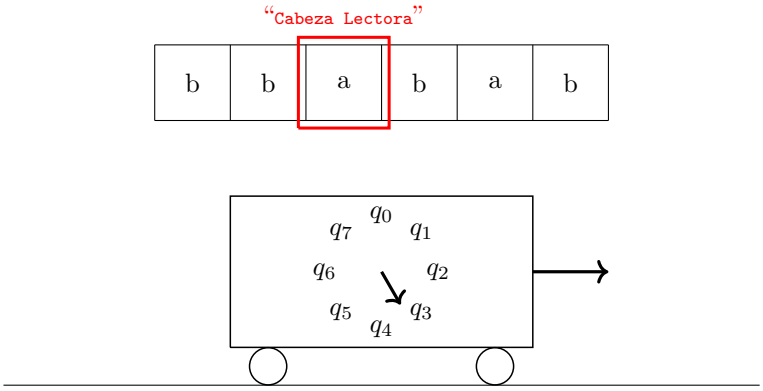


Figura 2.1: Es importante notar que un Autómata se mueve hacia *adelante* al contrario de una máquina de Turing `2.9`

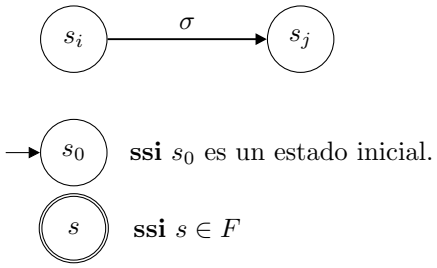
2.1.3. Representación

Tabla

En esta representación, el estado inicial se apunta con una **flecha**($\rightarrow$), los estados finales se **subrayan**.

k \ Σ	σ_0	σ_1	$\cdots$	σ_j	$\cdots$	σ_m
$\rightarrow s_0$				$\vdots$		
$\underline{s_1}$				$\vdots$		
$\vdots$				$\vdots$		
s_i	$\cdots$	$\cdots$	$\cdots$	s_k		
$\vdots$						
s_n						

Grafo



2.1.4. Palabra aceptada por M

Definición

Sea $M = (k, \Sigma, f, s_0, F)$ un AFD. Una función de estado terminal para M es la función:

$$\hat{f} : k \times \Sigma^* \rightarrow k$$

Sea $w \in \Sigma^*$ decimos que M **acepta** w *ssi* $\hat{f}(s_0, w) \in F$, caso contrario decimos que M **rechaza** w.

2.1.5. Lenguaje Aceptado por M

$$L(M) = \{w \in \Sigma^* / M \text{ acepta } w\}$$
$$L(M) = \{w \in \Sigma^* / \hat{f}(s_0, w) \in F\}$$

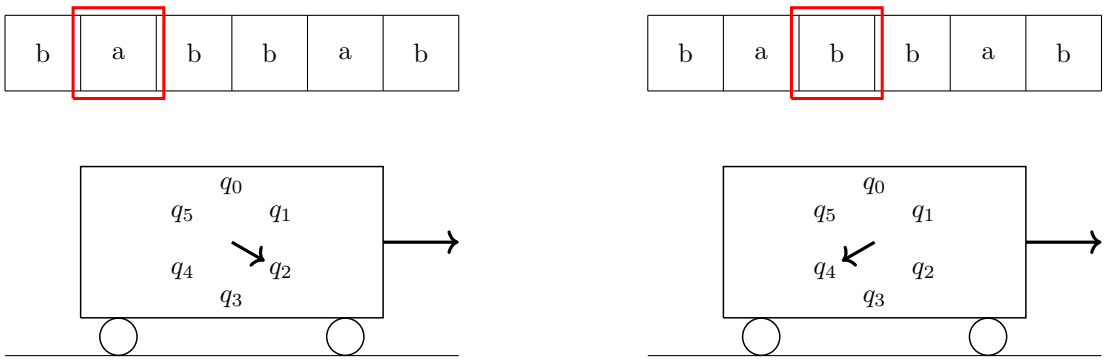
2.1.6. Configuración

Sea $M = (k, \Sigma, \delta, s_0, F)$ un AFD.

Una configuración de M es un elemento de $k \times \Sigma^*$
Podriamos de ir que una configuración es el resto de lo que nos falta procesar y un estado ("Lo que nos falta por leer").

Ejemplo(s)

- Sea las siguientes configuraciones: $(q_2, abbab)$ y $(q_4, bbab)$, les corresponde la siguientes gráficas respectivamente:



Con lo que decimos que una configuración es como una "instantánea" de un estado en un momento determinado.

Relación $\vdash_M$

Sea (q, w) y (q', w') dos configuraciones[1]:

$$(q, w) \vdash_M (q', w') \Leftrightarrow w = \sigma w' \text{ para algun } \sigma \in \Sigma \text{ y } \delta(q, \sigma) = q'$$

Denotamos por $\vdash_M^*$ a la clausura reflexiva transitiva de $\vdash_M$ y se lee "Conduce a" en cero o mas pasos (varios pasos).

Sea $M = (k, \Sigma, \sigma, s, F)$ un AFD y sea $w \in \Sigma^*$. Se dice que M acepta w **ssi**:

$$(s, w) \vdash_M^* (q, \lambda) \wedge q \in F$$

Lenguaje Aceptado por M (en términos de configuración)

$$L(M) = \{w \in \Sigma^* / M \text{ acepta } w\}$$
$$L(M) = \{w \in \Sigma^* / (s, w) \vdash_M^* (q, \lambda) \wedge q \in F\}$$

2.2. Autómata Finito no Determinístico (AFN)

2.2.1. Definición

Un autómata Finito no Deterministico (AFN) es una quintupla $M = (k, \Sigma, \Delta, s, F)$ donde:

- k : conjunto finito no vacio
- Σ : conjunto finito no vacio
- Δ : es un subconjunto finito de $k \times \Sigma^* \times k$
- $s \in k$
- $F \subseteq k$

2.2.2. Configuración

Una configuración es un elemento de $k \times \Sigma^*$.

Relación $\vdash_M$

Sean (q, w) y (q', w') dos configuraciones:

$$(q, w) \vdash_M (q', w') \Leftrightarrow w = uw' \text{ para algún } u \in \Sigma^* \text{ y } (q, u, q') \in \Delta$$

Denotamos por $\vdash_M^*$ la clausura reflexiva transitiva de $\vdash_M$ y se lee "Conduce a" en cero o mas pasos. Se dice que M acepta w **ssi**:

$$(s, w) \vdash_M^* (q, \lambda); q \in F$$

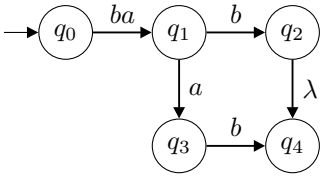
[1] $\vdash_M$ se lee *"conduce a"* en un paso.

Lenguaje Aceptado por M

$$L(M) = \{w \in \Sigma^* / M \text{ acepta } w\}$$
$$L(M) = \{w \in \Sigma^* / (s, w) \overset{*}{\underset{M}{\vdash}} (q, \lambda) \wedge q \in F\}$$

Ejemplo(s)

- Sea el siguiente autómata (**AFN**):



a) Procesar $w = baababaab$ (en términos de configuración).
$(q_0, baababaab) \overset{}{\underset{M}{\vdash}} (q_0, aababaab) \overset{}{\underset{M}{\vdash}} (q_0, ababaab) \overset{}{\underset{M}{\vdash}} (q_0, babaab) \overset{}{\underset{M}{\vdash}} (q_1, \lambda baab) \overset{}{\underset{M}{\vdash}} (q_4, baab) \overset{}{\underset{M}{\vdash}} (q_4, aab) \overset{}{\underset{M}{\vdash}} (q_4, ab) \overset{}{\underset{M}{\vdash}} (q_4, b) \overset{}{\underset{M}{\vdash}} (q_4, \lambda)$

$$\therefore (q_0, w) \overset{*}{\underset{M}{\vdash}} (q_4, \lambda) \wedge q_4 \in F \Leftrightarrow w \in L(M)$$

2.3. Equivalencia entre una AFD y un AFN

Teorema

Para cada AFN existe un AFD equivalente.

★ Prueba

Sea $M = (k, \Sigma, \Delta, s, F)$ un AFN

i.) Construimos $M' = (k', \Sigma, \Delta', s', F')$ eliminando todas las aristas de M que:

$$(q, u, q') \in \Delta \quad \wedge \quad |u| > 1$$

Si $u = \sigma_1\sigma_2 \ldots \sigma_k, k > 1$ entonces añadimos $p_1, p_2, \ldots, p_{k-1}$ estados y las nuevas transiciones:

$$(q, \sigma_1, p_1), (p_1, \sigma_2, p_2), \ldots, (p_{k-1}, \sigma_k, q')$$

a Δ para u tal que $|u| > 1$.

ii.) Construimos $M'' = (k'', \Sigma, \delta'', s'', F'')$
La idea clave es considerar que un AFN en un determinando instante se encuentra en un conjunto de estados:

- $k'' = \Sigma^{k'}$
- $F'' = \{Q \subseteq k' / Q \cap F' \neq \emptyset\}$

Formalmente:

$$E(q) = \{p \in k' / (q, \lambda) \overset{*}{\underset{M'}{\vdash}} (p, \lambda)\}$$

Equivalentemente:

$$E(q) = \{p \in k' / (q, w) \overset{*}{\underset{M'}{\vdash}} (p, w)\}$$

Donde:

- $s'' = E(s')$
- $\forall Q \subseteq k' \wedge$ para cada símbolo $\sigma \in \Sigma$

Ademas:

$$\delta''(Q,\sigma) = \cup\{E(p) : p \in k' \wedge (q,\sigma,p) \in \Delta', \exists q \in Q\}$$

Afirmamos que $\forall w \in \Sigma^*$ y $\forall p,q \in k'$:

$$(q,w) \vdash_{M'}^{*} (p,\lambda) \Leftrightarrow (E(q),w) \vdash_{M''}^{*} (P,\lambda)$$

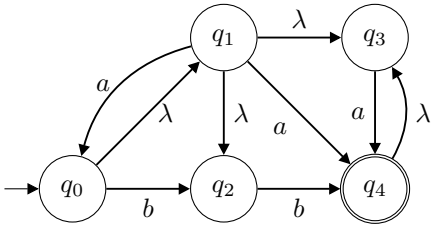
p.d. $M' \approx M''$
p.d. $L(M') = L(M'')$

$$\begin{aligned} w \in L(M') &\Leftrightarrow (s',w) \vdash_{M'}^{*} (q,\lambda), q \in F' \\ &\Leftrightarrow (E(s'),w) \vdash_{M''}^{*} (Q,\lambda) \\ &\Leftrightarrow (s'',w) \vdash_{M''}^{*} (Q,\lambda), Q \in F'' \\ &\Leftrightarrow w \in L(M'') \end{aligned}$$

$\therefore L(M') = L(M'')$

Ejemplo(s)

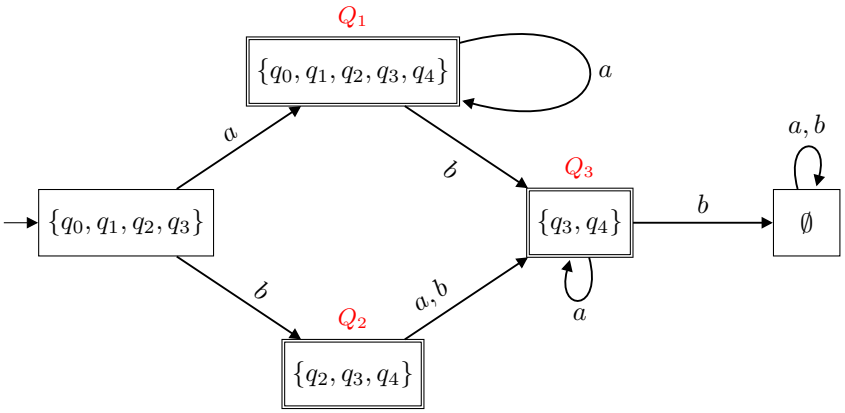
- Sea el autómata:



Realizamos su tabla de transiciones para realizar el algoritmo de Thompson:

Estados Viejos	Entrada λ	Entrada a	Entrada b
$\rightarrow q_0$	q_0, q_1, q_2, q_3	q_0, q_1, q_2, q_3, q_4	q_2, q_3, q_4
q_1	q_1, q_2, q_3	q_0, q_1, q_2, q_3, q_4	q_3, q_4
q_2	q_2	-	q_3, q_4
q_3	q_3	q_3, q_4	-
q_4	q_3, q_4	q_3, q_4	-

Finalmente tenemos el **AFD** equivalente:



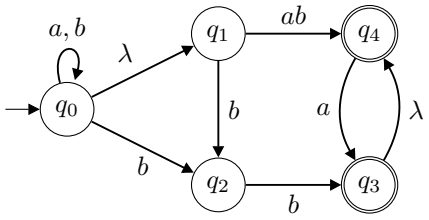
De este autómata podemos realizar una verificación de los estados finales con su **AFN**: $F = \{q_4\}$.

Para este **AFD**:

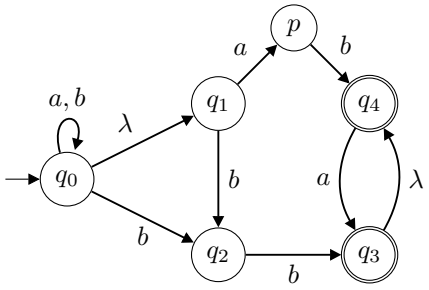
$$\begin{aligned} Q_1 \cap F &= \{q_4\} \neq \emptyset \\ Q_2 \cap F &= \{q_4\} \neq \emptyset \\ Q_3 \cap F &= \{q_4\} \neq \emptyset \end{aligned}$$

Por lo que decimos que este es su AFD equivalente.

- Sea el autómata M:



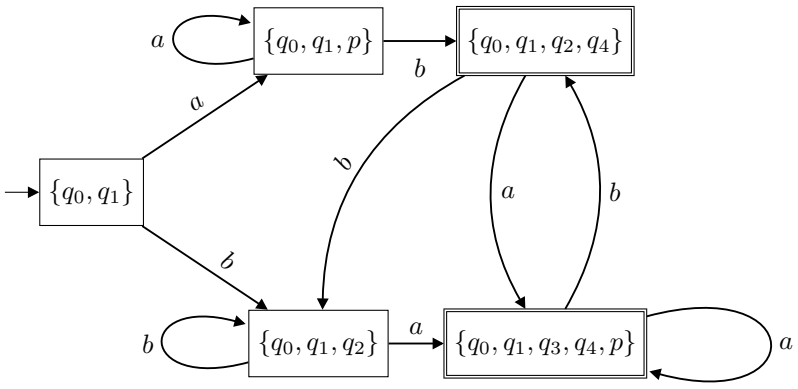
Tenemos M':



Lo hacemos para que solo procese símbolos y λ, añadiendo un estado. Notar que la cantidad de estados por añadir por cada palabra w es $|w|-1$. Realizamos entonces la tabla para luego realizar el **AFD** equivalente:

Estados Viejos	Entrada λ	Entrada a	Entrada b
$\rightarrow q_0$	q_0, q_1	q_0, q_1, p	q_0, q_1, q_2
q_1	q_1	p	q_2
q_2	q_2	q_3, q_4	-
q_3	q_3, q_4	q_3, q_4	-
q_4	q_4	q_3, q_4	-
p	p	-	q_4

Finalmente tenemos el **AFD** equivalente:



2.4. Propiedades de los Lenguajes Aceptados por AF's

Teorema

La clase de Lenguajes aceptados por AF's es cerrada bajo la:

- Unión
- Concatenación

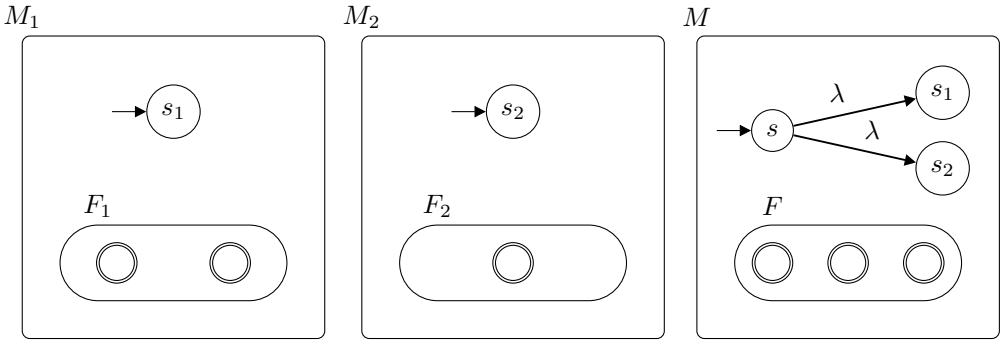
- Estrella de Kleene
- Complementación
- Intersección

Prueba

Sean $L(M_1)$ y $L(M_2)$ lenguajes aceptados por $M_1 = (k_1, \Sigma, \Delta_1, s_1, F_1)$ y $M_1 = (k_2, \Sigma, \Delta_2, s_2, F_2)$:

a) Unión

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1) \cup L(M_2)$

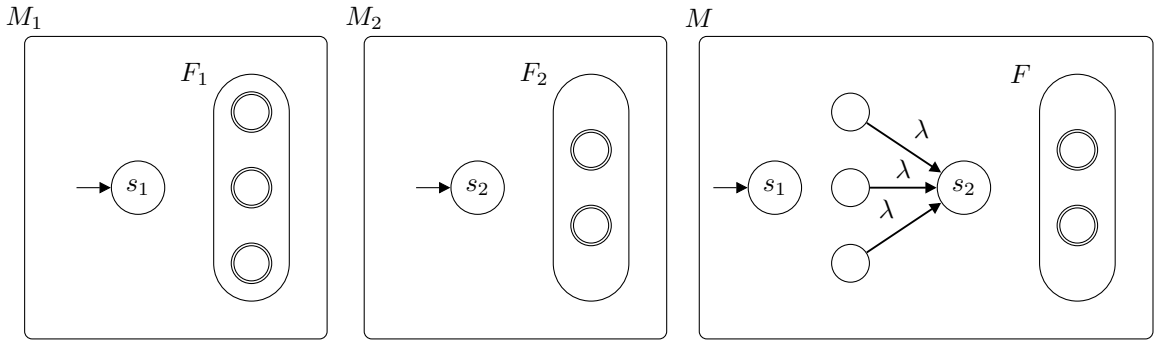


Donde:

- $k = k_1 \cup k_2 \cup \{s\}$
  donde s es un nuevo estado (inicial).
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(s, \lambda, s_1), (s, \lambda, s_2)\}$
- s : nuevo estado añadido
- $F = F_1 \cup F_2$

b) Concatenación

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1)L(M_2)$



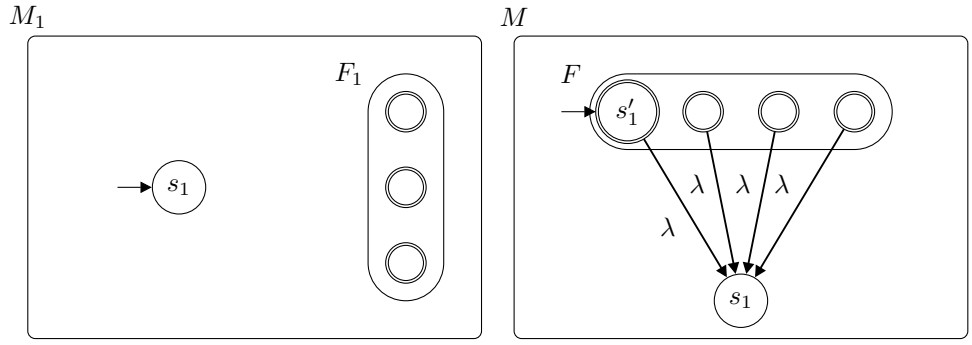
Donde:

- $k = k_1 \cup k_2$
- $\Delta = \Delta_1 \cup \Delta_2 \cup (F_1 \times \{\lambda\} \times \{s_2\})$
- $s = s_1$
- $F = F_2$

Los estados finales del primer autómata ya no son finales, en cambio sacamos aristas λ al estado inicial del segundo autómata.

c) Estrella de Kleene

Construimos $M = (k, \Sigma, \Delta, s, F)$ tal que: $L(M) = L(M_1)^*$



Donde:

- $k = k_1 \cup \{s'_1\}$
  donde s'_1 es un nuevo estado (*inicial y terminal*).
- $\Delta = \Delta_1 \cup (F \times \{\lambda\} \times \{s_1\})$
- $s = s'_1$
- $F = F_1 \cup \{s'_1\}$

d) Complementación

Sea $M = (k, \Sigma, \delta, s, F)$ un **AFD**.
Donde:

$$\Sigma^* - L(M) \text{ es aceptado por } \overline{M} = (k, \Sigma, \delta, s, k - F)$$

e) Intersección

Sea:

$$\begin{aligned} L_1 \cap L_2 &= \overline{\overline{L_1 \cap L_2}} \\ &= \overline{\overline{L_1} \cup \overline{L_2}} \\ &= \Sigma^* - \underbrace{\underbrace{[\underbrace{(\Sigma^* - L_1)}_{1} \cup \underbrace{(\Sigma^* - L_2)}_{2}]}_{3}}_{4} \end{aligned}$$

Notese que en $1, 2, 3, 4$ se puede ver que en cada operación al aplicarla se obtiene otro AF que puede ser nuevamente usada en la siguiente operación y así sucesivamente.

Teorema

Existen algoritmos para responder las siguientes preguntas acerca de Automatas Finitos:

a) Dado un AF M y una palabra w, ¿$w \in L(M)$?

b) Dado un AF M, es ¿$L(M) = \emptyset$?

c) Dado un AF M, es ¿$L(M) = \Sigma^*$?

d) Dado 2 AF's M_1 y M_2 ,es ¿$L(M_1) \subseteq L(M_2)$?

e) Dado 2 AF's M_1 y M_2 ,es ¿$L(M_1) = L(M_2)$?

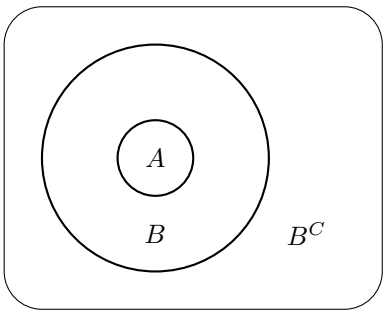
Prueba

a) Esto es lo que ya hemos ido haciendo.

b) Simplemente nos basta analizar si el automata no acepta ninguna palabra.

c) $M \to L(\bar{M}) = \emptyset$ (Basta realizar el complemento del AF y ver que el lenguaje aceptado sea $\emptyset$). **(b)**

d) $A \subseteq B \Leftrightarrow A \cap B^C = \emptyset$



$$L(M_1) \subseteq L(M_2) \Leftrightarrow L(M_1) \cap [\Sigma^* - L(M_2)] = \emptyset$$

e) Realizamos la doble inclusión:

$$L(M_1) \subseteq L(M_2) \wedge L(M_2) \subseteq L(M_1) \textbf{ (d)}$$

2.5. Automatas Finitos y Expresiones Regulares

Teorema

Un Lenguaje es regular ssi es aceptado por un Automata Finito.

Prueba

i.) Supongamos que la propiedad se cumple para γ tal que $|\gamma| < n$.

Para $|\gamma| = n$

- $\gamma = \alpha \cup \beta$

$$\begin{aligned} L(\gamma) = L(\alpha \cup \beta) &= L(\alpha) \cup L(\beta) \\ &= L(M_1) \cup L(M_2) = L(M) \end{aligned}$$

- $\gamma = \alpha\beta$

$$\begin{aligned} L(\gamma) = L(\alpha\beta) &= L(\alpha)L(\beta) \\ &= L(M_1)L(M_2) = L(M) \end{aligned}$$

- $\gamma = \alpha^*$

$$\begin{aligned} L(\gamma) = L(\alpha^*) &= L(\alpha)^* \\ &= L(M_1)^* = L(M) \end{aligned}$$

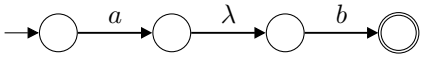
Ejemplo(s)

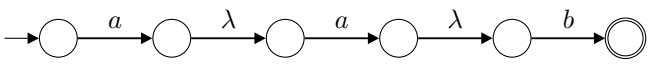
- Dada la **ER** $\alpha = (ab \cup aab)^*$ hallar un autómata M tal que: $L(M) = L(\alpha)$.

  El lenguaje generado por α sería: $L(\alpha) = \{\lambda, ab, aab, abaab, abab, \ldots\}$

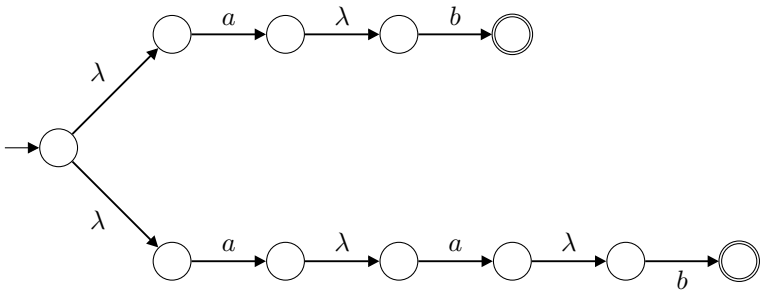
Paso 1: a, b

Paso 2: ab, aab

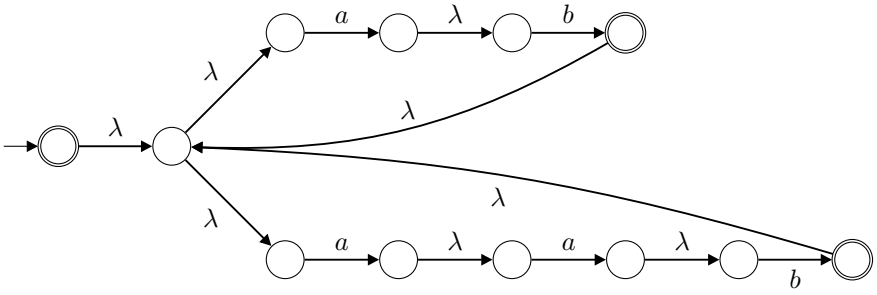




Paso 3: $ab \cup aab$



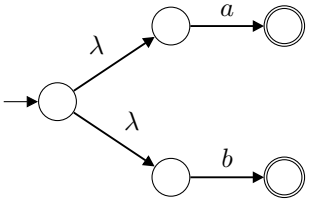
Paso 4: $(ab \cup aab)^*$



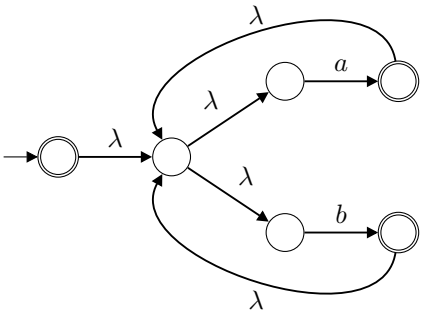
- Dada la **ER** $\alpha = b(a \cup b)^* b$ hallar un autómata M tal que: $L(M) = L(\alpha)$.

Paso 1: a, b

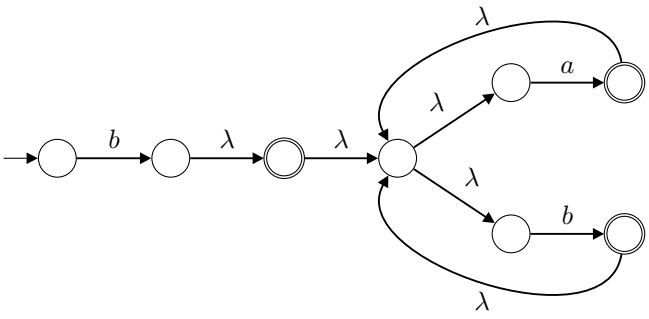
Paso 2: $a \cup b$



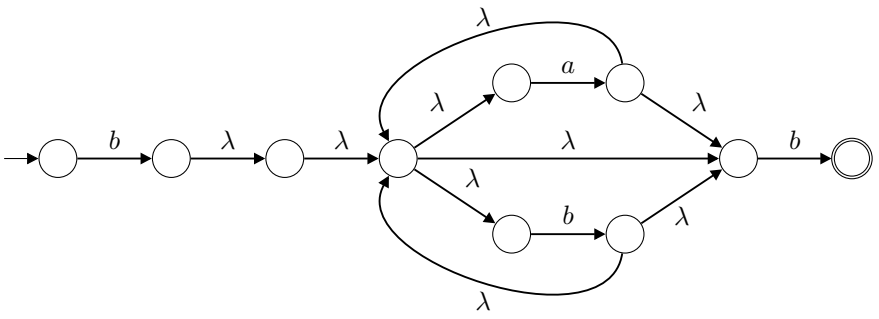
Paso 3: $(a \cup b)^*$



Paso 4: $b(a \cup b)^*$



Paso 5: $b(a \cup b)^* b$



ii.) Si un lenguaje es aceptado por un AF entonces es un lenguaje regular.

Prueba

Sea $M = (K, \Sigma, \delta, s, F)$ el **AFD** que acepta $L(M)$:

p.d. $R = L(M)$

Representaremos $L(M)$ como la unión de muchos (pero en número finito) lenguajes simples.

Sea $k = \{q_1, q_2, q_3, \ldots, q_n\}, s = q_1$ para $i, j = 1, 2, \ldots, n; k = 1, 2, \ldots, n+1$

Definimos $R(i, j, k)$ como el conjunto de todas las palabras que nos llevan de q_i a q_j sin pasar por estados con subíndice k o mayores.

Formalmente

$$R(i,j,k) = \{x \in \Sigma^* / (q_i, x) \overset{*}{\underset{M}{\vdash}} (q_j, \lambda) \wedge (q_i, x) \overset{*}{\underset{M}{\vdash}} (q_l, y) \text{ entonces } l < k \vee (y = \lambda \wedge l = j) \vee (y = x \wedge l = i)\}$$

Si $k = n + 1$:[2]

$$R(i,j,n+1) = \{x \in \Sigma^* / (q_i, x) \overset{*}{\underset{M}{\vdash}} (q_j, \lambda)\}$$

Esto es:

$$L(M) = \cup\{R(1, j, n+1), q_j \in F\}$$

p.d. $R(i, j, k)$ son regulares (inducción) $k = 1$

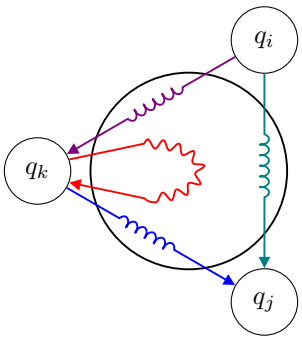
$$R(i,j,1) = \begin{cases} \{\sigma \in \Sigma / \delta(q_i, \sigma) = q_j\}, i \neq j \\ \\ \{\lambda\} \cup \{\sigma \in \Sigma / \delta(q_i, \sigma) = q_j\}, i \neq j \end{cases}$$

Son regulares ya que $R(i, j, 1)$ son **finitos**.

HI
Para $k = 1, 2, \ldots, n$ $R(i, j, k)$ son Regulares.

p.d. Para $k + 1$ es decir $R(i, j, k + 1)$ es regular:

[2]Notese que no es posible llegar a $n + 1$ ya que k tiene n estados



$$R(i,j,k+1) = \underbrace{R(i,j,k)}_{Regular} \cup \underbrace{\underbrace{R(i,k,k)}_{Regular}\underbrace{R(k,k,k)^*}_{Regular}\underbrace{R(k,j,k)}_{Regular}}_{Regular}$$
$$Regular$$

$R(i,j,k)$ es regular por la clausura de lenguaje regular respecto a la unión, concatenación y Estrella de Kleene.

2.6. Lenguajes no Regulares

2.6.1. Teoría

Sea L un lenguaje regular infinito. Entonces existen palabras x, y, z tales que $y \neq \lambda$ y $xy^n z \in L$. Para cada $n \geq 0$.

Ejemplo(s)

- Demostrar que:

$$L = \{a^n b^n : n \geq 0\} \text{ es } \textbf{no regular.}$$

Prueba (RAA)

$$L \text{ es regular infinito.}$$

- **Caso** 1: Si y consta exclusivamente de "a"s.

$$x = a^p, y = a^q, z = a^r b^s$$

$$s, q > 0 \quad p, r \geq 0$$

$$\begin{aligned} xy^n z &\in L \\ a^p (q^q) a^r b^s &\in L \\ a^{p+qn+r} b^s &\in L \end{aligned}$$

$$\begin{aligned} n = 0 &\quad \underbrace{aa}_{x}\underbrace{abbbbbb}_{z} \\ \boxed{\text{n = 1}} &\quad \underbrace{aa}_{x}\underbrace{aaa}_{y}\underbrace{abbbbbb}_{z} \\ n = 2 &\quad \underbrace{aa}_{x}\underbrace{aaaaaa}_{y}\underbrace{abbbbbb}_{z} \end{aligned}$$

Solo para $n = 1$. Tenemos una contradicción. $\Rightarrow\Leftarrow$.

- **Caso** 2: Si y consta exclusivamente de "b's.

Tendriamos crecimiento de "b's.

- **Caso** 3: Si y consta de "a"s y "b"s.

Tendriamos intercalado.

$\therefore$ L no es regular.

- Demostrar que:

$$L = \{a^n : n \text{ es primo}\} \text{ es } \textbf{no regular.}$$

Prueba (RAA)

Supongamos L es regular infinito.

$$x = a^p, y = a^q, z = a^r$$
$$q > 0 \quad p, r \geq 0$$

$$\begin{aligned} xy^n z &\in L, \forall n \geq 0 \\ a^p (q^q)^n a^r &\in L, \forall n \geq 0 \\ a^{p+qn+r} &\in L, \forall n \geq 0 \end{aligned}$$

$$n = p + 2q + r + 2$$

$$\begin{aligned} p + q(p + 2q + r + 2) + r &= p + qp + 2q^2 + qr + 2q \\ &= p(q+1) + 2q(q+1) + r(q+1) \\ &= \underbrace{(q+1)}_{\heartsuit}(p + 2q + r) \end{aligned}$$

$\heartsuit$ q debe ser cuanto mínimo 1 pero tenemos $q + 1$, por lo cual no puede ser, ya que no cumpliría con un caso.

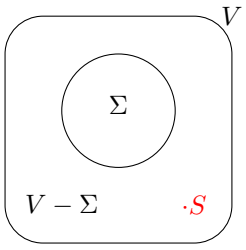
$\therefore$ L no es regular.

2.7. Gramáticas

2.7.1. Definición

Una gramática Libre de Contexto es una cuádrupla $G = (V, \Sigma, R, S)$ donde:

- G es un alfabeto.
- Σ es un subconjunto de V (Conjunto de Símbolos Terminales)
- $S \in (V - \Sigma)$ (Símbolo Inicial)
- R es un conjunto finito de $(V - \Sigma) \times V^*$ (Reglas)



A los elementos de $V - \Sigma$ se les llama *símbolos no terminales.*

Notación

Para cada $A \in V - \Sigma$, $w \in \Sigma^*$

$$(A, w) \in R \Leftrightarrow A \underset{G}{\longrightarrow} w$$

Donde G se refiere a la gramática.

Derivación

Para cada par de palabras $u, v \in V^*$ escribimos $u \underset{G}{\Rightarrow} v$ ssi existen palabras $x, y, v' \in V^*, A \in (V - \Sigma)$ tales que:

$$u = xAy, \quad v = xv'y, \quad A \underset{G}{\longrightarrow} v'$$

Se denota por $u \overset{*}{\underset{G}{\Rightarrow}} v$ la clausura reflexiva transitiva de $\underset{G}{\Rightarrow}$

Lenguaje Generado por G

$$L(G) = \left\{ w \in \Sigma^* / S \overset{*}{\underset{G}{\Rightarrow}} w \right\}$$

Un lenguaje es libre de contexto **ssi** existe una gramatica libre de contexto que genera el Lenguaje.

Ejemplo(s)

- Un ejemplo de una gramática: $\alpha = a(a^* \cup b^*)b$.

$$S \to aMb$$

$$\begin{array}{ll} M \to A & M \to B \\ A \to \lambda & B \to \lambda \\ A \to aA & B \to bB \end{array}$$

- Del anterior ejemplo, esta sería la notación: $G = (V, \Sigma, R, S)$ donde:
  - $V = \{S, M, A, B, a, b\}$
  - $\Sigma = \{a, b\}$
  - $S \in V - \Sigma$
  - $R = \{(S, aMb), (M, A), (M, B), (A, \lambda), (B, \lambda), (B, bB), (A, aA)\}$
- Del ejemplo anterior podemos ver algunas **ER** generadas:
  - $S \Rightarrow \underline{a}M\underline{b} \Rightarrow \underline{a}B\underline{b} \Rightarrow \underline{ab}B\underline{b} \Rightarrow \underline{ab}\lambda\underline{b} \Rightarrow abb$

    $\therefore S \overset{*}{\Rightarrow} abb, abb \in L(G)$
  - $S \Rightarrow \underline{a}M\underline{b} \Rightarrow \underline{aa}A\underline{b} \Rightarrow \underline{aaa}A\underline{b} \Rightarrow \underline{aaaa}A\underline{b} \Rightarrow \underline{aaaa}\lambda\underline{b} \Rightarrow aaaab$

    $\therefore S \overset{*}{\Rightarrow} aaaab, aaaab \in L(G)$

2.7.2. Gramáticas Regulares

Definición

Una Gramática Libre de Contexto $G = (V, \Sigma, R, S)$ es regular ssi:

$$R \subseteq (V - \Sigma) \times \Sigma^*((V - \Sigma) \cup \{\lambda\})$$

Teorema

Un lenguaje es Regular ssi es generada por una Gramática Regular:

Prueba

$\Rightarrow$

Sea $M = (K, \Sigma, \sigma, s, F)$ construimos $G = (V, \Sigma, R, S)$ donde:

- $V = K \cup \Sigma$
- $S = s$
- $R = \{q \to ap : \delta(p, a) = p\} \cup \{q \to \lambda : q \in F\}$

$\Leftarrow$

Sea $G = (V, \Sigma, R, S)$ construimos $M = (K, \Sigma, \Delta, s, F)$ donde:

- $K = (V - \Sigma) \cup \{f\}$ donde f es un nuevo elemento.
- $s = S$
- $F = \{f\}$
- $\Delta = \{(A, w, B); A \to wB, A, B \in (V - \Sigma), w \in \Sigma^*\} \cup \{(A, w, f); A \to w, A \in (V - \Sigma), w \in \Sigma^*\}$

Ejemplo(s)

- $G = (V, \Sigma, R, S)$ donde:

$$V = \{S, A, B, a, b\} \qquad \Sigma = \{a, b\}$$

$$R = \{R \to bA, S \to aB, A \to abaS, B \to babS, S \to A\}$$

Como un pequeño concejo, podemos decir que para que sea *regular*, las mayúsculas deben estar al extremo derecho o estar compuesto de minúscula únicamente.

2.8. Autómata con Pila

Definición

Un autómata con Pila es una sextupla:

$$M = (K, \Sigma, \Gamma, \Delta, s, F)$$

donde :

- k : Conjunto Finito no vacío.
- Σ : Símbolos de Entrada
- Γ : es un conjunto finito no vacío *(llamado alfabeto de la Pila).*
- $s \in K$
- $F \subseteq K$
- Δ es un subconjunto finito de $(K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$ (Relación de Transición)

Significado de una Transición

$$((p, u, \beta), (q, \gamma)) \in \Delta$$

"Estando en p leyendo w pasa al estado q y reemplaza β por γ."

Empujar a

$$((p, u, \lambda), (q, \alpha)) \in \Delta$$

Sacar de

$$((p, u, \alpha), (q, \lambda)) \in \Delta$$

2.8.1. Configuración

Una configuración es un elemento de:

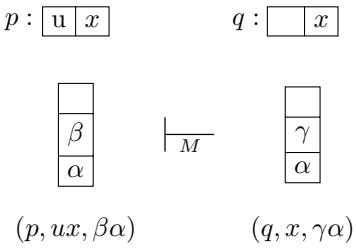
$$K \times \Sigma^* \times \Gamma^*$$

La relación $\vdash_M$

Para cada $((p,u,\beta),(q,\gamma)) \in \Delta$ y para cada $x \in \Sigma^*, \alpha \in \Gamma^*$ definimos:

$$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha)$$

Para ilustrar un poco mejor la idea tenemos la siguiente gráfica:



$(p, ux, \beta\alpha)$ $(q, x, \gamma\alpha)$

Palabra aceptada por M

Sea $w \in \Sigma^*$:

$$M \text{ acepta } w \Leftrightarrow (s, w, \lambda) \vdash_M^* (q, \lambda, \lambda); q \in F$$

Lenguaje aceptado por M

$$L(M) = \{w \in \Sigma^* / (s, w, \lambda) \vdash_M^* (q, \lambda, \lambda); q \in F\}$$

2.8.2. Autómatas con Pila y Gramáticas Libre de Contexto

Definición

La clase de lenguajes aceptados por autómatas con pila es exactamente la clase de lenguajes libres de contexto:

Prueba $\Leftarrow$

Sea $G = (V, \Sigma, R, S)$ construimos $M = (K, \Sigma, \Gamma, \Delta, s, F)$ tal que $L(M) = L(G)$ donde:

$$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$$

Transiciones

$$((p, \lambda, \lambda), (q, S)) \tag{2.1}$$
$$((q, \lambda, A), (q, x)); \forall A \to x \in R \tag{2.2}$$
$$((q, a, a), (q, \lambda)); \forall a \in \Sigma \tag{2.3}$$

2.8.3. Propiedades de los Lenguajes Libres de Contexto

Teorema

Los lenguajes libres de contexto son cerrados bajo la unión, concatenación y estrella de Kleene.

Prueba

Sean $G_i = (V_i, \Sigma_i, R_i, S_i) \quad i = 1, 2$ las Gramáticas Libres de Contexto y sin perdida de generalidad asumimos que $(V_1 - \Sigma_1)$ y $(V_2 - \Sigma_2)$ son disjuntos.

- **Unión**

  S es un nuevo símbolo

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}, S)$$

- **Concatenación**

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S \to S_1 S_2\}, S)$$

- **Estrella de Kleene**

$$G = (V_1, \Sigma_1, R_1 \cup \{S_1 \to \lambda, S_1 \to S_1 S_1\}, S_1)$$

Teorema

La intersección de un L.L.C con un lenguaje regular es un lenguaje libre de Contexto.

Teorema de Pumping

Sea G una G.L.C. entonces existe un número k que depende de G tal que cualquier palabra $w \in L(G)$ de longitud $> k$, se puede reescribir $w = uvxyz$ de tal manera que ya sea v o y es no vacía y $uv^n xy^n z \in L(G)$. **Para cada** $n \geq 0$.

Teorema

Los L.L.C. no son cerrados bajo la complementación e intersección.

2.9. Máquinas de Turing

Definición

Una máquina de Turing es una cuadrupla $M = (K, \Sigma, \delta, s)$ donde:

- K : Conjunto Finito no Vacío , $h \notin K$
- Σ : Conjunto Finito no Vacío , $\# \in \Sigma, L, R \notin \Sigma$
- $s \in K$
- $\delta : K \times \Sigma \to (K \cup \{h\}) \times (\Sigma \cup \{L, R\})$

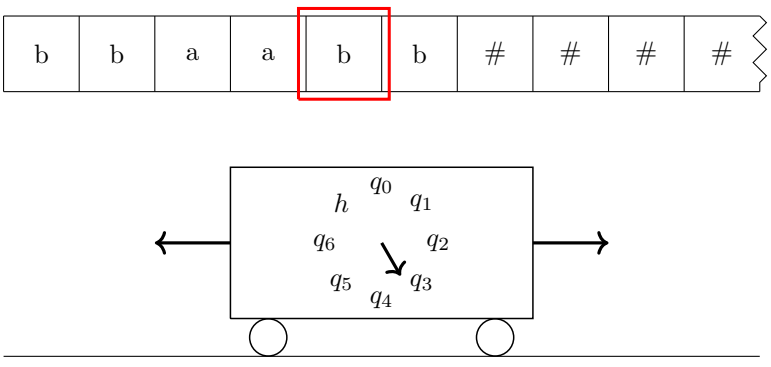
i.) Si $b \in \Sigma$ la máquina reescribe b por a.

ii.) Si $b = L$ la cabeza lectora se mueve a la izquierda.

iii.) Si $b = R$ la cabeza lectora se mueve a la derecha.

Si la máquina esta en h se dice que la máquina se *detiene*.

Interpretación



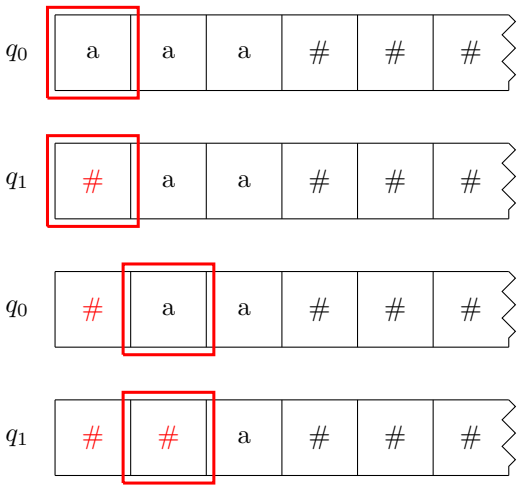
En una Máquina de Turing la *cinta* es infinita a la derecha. Donde # es el símbolo en blanco.

Ejemplo(s)

- Sea la siguiente tabla de transiciones de una Máquina de Turing:

	a	$\#$
$\underline{q_0}$	$(q_1, \#)$	$(h, \#)$
q_1	(q_0, a)	(q_0, R)

$S = q_0$ Procesar: aaa.



Teorema

Los siguientes modelos de máquinas de Turing son igualmente poderosos:

- La Máquina de Turing Estándar.
- La Máquina de Turing de varias cabezas.
- La Máquina de Turing de varias pistas.
- La Máquina de Turing de varias cintas.
- La Máquina de Turing Bidimensional.
- Automata con Pila con 2 Pilas
- IBM Mainframe